

INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DE SÃO PAULO
CAMPUS CAMPINAS

RONALD MATEUS JESUS FLORENCIO DE ALMEIDA

DOEAQUI: SISTEMA WEB PARA LOCALIZAR CENTROS DE ARRECADAÇÃO

CAMPINAS - SP

2022

RONALD MATEUS JESUS FLORENCIO DE ALMEIDA

DOEAQUI: SISTEMA WEB PARA LOCALIZAR CENTROS DE ARRECADAÇÃO

Trabalho de Conclusão de Curso apresentado como exigência parcial para obtenção do diploma do Curso Tecnologia em Análise e Desenvolvimento de Sistemas do Instituto Federal de Educação, Ciência e Tecnologia Câmpus Campinas.

Orientador: Prof. Dr. Andreiwid Sheffer
Correa.

CAMPINAS - SP

2022

Ficha catalográfica
Instituto Federal de São Paulo – Câmpus Campinas
Biblioteca
Rosana Gomes – CRB 8/8733

Almeida, Ronald Mateus Jesus Florêncio de
A447d DoeAqui: sistema web para localizar centros de arrecadação / Ronald Mateus
Jesus Florêncio de Almeida. – Campinas, SP: [s.n.], 2022.
51 f. il. :

Orientador: Andreiwid Sheffer Correa
Trabalho de Conclusão de Curso (graduação) – Instituto Federal de
Educação, Ciência e Tecnologia de São Paulo Câmpus Campinas. Curso de
Tecnologia em Análise e Desenvolvimento de Sistemas, 2022.

1. Fome. 2. Doações. 3. Alimentos. 4. Sites da web. I. Instituto Federal de
Educação, Ciência e Tecnologia de São Paulo Câmpus Campinas. Curso de
Tecnologia em Análise e Desenvolvimento de Sistemas. II. Título.

ATA N.º 17/2022 - TADS-CMP/DAE-CMP/DRG/CMP/IFSP

Ata de Defesa de Trabalho de Conclusão de Curso - Graduação

Na presente data, realizou-se a sessão pública de defesa do Trabalho de Conclusão de Curso intitulado **DoeAqui: Sistema Web para localizar centros de arrecadação**, apresentado pelo aluno **Ronald Mateus Jesus Florencio de Almeida (CP3007758)** do Curso **SUPERIOR DE TECNOLOGIA EM ANÁLISE E DESENVOLVIMENTO DE SISTEMAS**(Câmpus Campinas). Os trabalhos foram iniciados às **18h00** pelo(a) Professor(a) presidente da banca examinadora, constituída pelos seguintes membros:

Membros	Instituição	Presença (Sim/Não)
ANDREIWID SHEFFER CORRÊA (Presidente/Orientador)	IFSP-CMP	SIM
JOSÉ AMÉRICO DOS SANTOS MENDONÇA (Examinador 1)	IFSP-CMP	SIM
APARECIDO DONISETE PIRES DE MORAES (Examinador 2)	IFSP-CMP	SIM

Observações:

A banca examinadora, tendo terminado a apresentação do conteúdo da monografia, passou à arguição do(a) candidato(a). Em seguida, os examinadores reuniram-se para avaliação e deram o parecer final sobre o trabalho apresentado pelo(a) aluno(a), tendo sido atribuído o seguinte resultado:

Aprovado(a) Reprovado(a)

Proclamados os resultados pelo presidente da banca examinadora, foram encerrados os trabalhos e, para constar, eu lavrei a presente ata que assino em nome dos demais membros da banca examinadora.

Câmpus Campinas, 11 de novembro de 2022

Documento assinado eletronicamente por:

- Andreiwid Sheffer Correa, PROFESSOR ENS BASICO TECN TECNOLOGICO, em 11/11/2022 18:59:51.
- Aparecido Donisete Pires de Moraes, PROF ENS BAS TEC TECNOLOGICO-SUBSTITUTO, em 14/11/2022 09:36:26.
- Jose Americo dos Santos Mendonca, PROFESSOR ENS BASICO TECN TECNOLOGICO, em 14/11/2022 10:57:49.

Este documento foi emitido pelo SUAP em 11/11/2022. Para comprovar sua autenticidade, faça a leitura do QRCode ao lado ou acesse <https://suap.ifsp.edu.br/autenticar-documento/> e forneça os dados abaixo:

Código Verificador: 446451
Código de Autenticação: d27fd0d9e7



ATA N.º 17/2022 - TADS-CMP/DAE-CMP/DRG/CMP/IFSP

Dedico esse trabalho aos meus familiares, minha namorada, colegas, amigos, professores e orientadores que contribuíram na minha formação acadêmica.

AGRADECIMENTOS

Primeiramente agradeço aos meus pais, por sempre buscarem me proporcionar a melhor educação possível, mesmo em tempos difíceis.

Agradeço a todos os professores e servidores do Instituto Federal de Educação, Ciência e Tecnologia de São Paulo Campus Campinas, por todos os ensinamentos, conselhos, paciências e empatia.

Agradeço a todos os colegas de classe e amigos que contribuíram na criação de momentos inesquecíveis durante minha jornada acadêmica.

Agradeço ao meu primeiro orientador Alencar de Melo e ao orientador Andreiwid Sheffer, por todas as dicas, conselhos, incentivos e ajuda.

Por último, mas não menos importante, quero agradecer a minha namorada que sempre esteve ao meu lado, proporcionando apoio e amor incondicional.

“Você deve aproveitar os pequenos desvios. Ao máximo. Porque é onde você encontrará coisas mais importantes do que aquilo que você realmente deseja.”

Ging Freecss

RESUMO

A Rede Brasileira de Pesquisa em Soberania e Segurança Alimentar e Nutricional indica que, em 2022, a insegurança alimentar grave atingiu 15,5% da população brasileira, ou seja, 33,1 milhões de pessoas. Levando esse fato em consideração, este trabalho aplica conceitos de mapeamento regional, para facilitar o acesso e localização de pontos de arrecadação e distribuição de alimentos. Para isso, foi desenvolvido um *website*, que localiza os pontos de doação mais próximos do usuário e facilita o gerenciamento do estoque das doações. Desse modo, pessoas que estão dispostas a doar podem encontrar um local próximo, assim como, os produtos que o local necessita receber e aqueles que serão beneficiados podem se dirigir ao centro mais perto.

O sistema apresentou resultados positivos, mostrou-se eficiente na localização e no gerenciamento do ponto de doação.

Palavras-chave: Doação; Mapeamento; Sistema *Web*; Fome.

ABSTRACT

The Brazilian Research Network on Food and Nutritional Sovereignty and Security indicates that, in 2022, severe food insecurity reached 15.5% of the Brazilian population, that is, 33.1 million people. Taking this fact into account, this work applies regional mapping concepts to facilitate access and location of food collection and distribution points. For this, a website was developed, which locates the donation points closest to the user and facilitates the management of the donation stock. In this way, people who are willing to donate can find a nearby location, as well as the products that the location needs to receive and those who will benefit from it can go to the nearest center.

The system showed positive results, it proved to be efficient in locating and managing the donation point.

Keywords: Donation; Mapping; Web System; Hungry.

LISTA DE FIGURAS

Figura 1 – Gráfico sobre a segurança alimentar no Brasil em milhões de pessoas.....	16
Figura 2 – Exemplo de URL.....	23
Figura 3 – Diagrama de Caso de Uso.....	27
Figura 4 – Prototipação de telas com o Figma.....	28
Figura 5 – Organização estrutural do sistema.....	28
Figura 6 – Exemplo de código do <i>model</i> da criação de um centro.....	29
Figura 7 – Exemplo de código do <i>controller</i> da criação de um centro.....	30
Figura 8 – Exemplo de código da <i>view</i> da criação de um centro.....	31
Figura 9 – Exemplo de código das <i>routes</i> do sistema.....	32
Figura 10 – Visão geral do MVC.....	33
Figura 11 – Documentos do banco de dados.....	34
Figura 12 – Visão do administrador do Firebase <i>Authentication</i>	34
Figura 13 – Visão do administrador do documento dos centros.....	35
Figura 14 – Visão do administrador do documento dos estoques.....	36
Figura 15 – Função de localização do ponto mais próximo.....	37
Figura 16 – Sistema hospedado com Heroku.....	38
Figura 17 – Tela Inicial.....	39
Figura 18 – Tela de identificação.....	40
Figura 19 – Tela de acesso negado.....	41
Figura 20 – Tela de criação de conta do gerente.....	41
Figura 21 – Tela de recuperar senha.....	42
Figura 22 – E-mail contendo o link de redirecionamento.....	42
Figura 23 – Página de redefinição da senha.....	43
Figura 24 – Tela de gerenciamento.....	43
Figura 25 – Tela de visualização do ponto de doação.....	44
Figura 26 – Tela de criação do ponto de doação.....	45
Figura 27 – Tela de visualização do estoque.....	46
Figura 28 – Tela de controle do estoque.....	47
Figura 29 – Tela de configurações do perfil.....	48
Figura 30 – Tela de história do website.....	48
Figura 31 – Tela de informações de contato.....	49

LISTA DE TABELAS

Tabela 1 – Principais características de sites de arrecadação..... 25

LISTA DE SIGLAS

HTML	HyperText Markup Language
CSS	Cascading Style Sheets
API	Application Programming Interface
IDE	Integrated Development Environment
VS CODE	Visual Studio Code
JSON	JavaScript Object Notation

SUMÁRIO

1. INTRODUÇÃO.....	14
2. JUSTIFICATIVA.....	15
3. OBJETIVOS.....	17
3.1. Objetivo geral.....	17
3.2. Objetivos específicos.....	17
4. FUNDAMENTAÇÃO TEÓRICA.....	18
4.1. Sistemas web.....	18
4.1.1. Back-end.....	18
4.1.2. Front-end.....	19
4.2. Testes e usabilidade.....	19
4.3. Engenharia de requisitos.....	20
4.4. Arquitetura de software.....	21
4.5. Geolocalização.....	21
4.6. Bibliotecas e frameworks.....	22
4.7. Rotas do sistema.....	22
4.8. Template engine.....	23
4.9. Firebase Firestore e Firebase Authentication.....	23
4.10. Ferramentas e tecnologias utilizadas.....	24
5. METODOLOGIA.....	25
5.1. Análise de requisitos.....	25
5.1.1. Requisitos funcionais.....	26
5.1.2. Requisitos não funcionais.....	26
5.2. Estudo de caso de uso.....	27
5.3. Criação do layout.....	27
5.4. Arquitetura e rotas do sistema.....	28
5.4.1. Model.....	29
5.4.2. Controller.....	29

5.4.3. View.....	30
5.4.4. Routes.....	31
5.4.5. Visão geral.....	32
5.5. Banco de dados e autenticação.....	33
5.5.1. Gerente.....	34
5.5.2. Centro e estoque.....	35
5.6. Localização do ponto mais próximo.....	36
5.7. Testes.....	37
5.8. Hospedagem da aplicação.....	37
6. RESULTADOS.....	39
6.1. Telas do sistema.....	39
6.1.1. Tela inicial.....	39
6.1.2. Tela de identificação.....	40
6.1.3. Tela de acesso negado.....	40
6.1.4. Tela de criar conta do gerente.....	41
6.1.5. Tela de recuperar senha.....	42
6.1.6. Tela de gerenciamento.....	43
6.1.7. Tela de visualização do ponto de doação.....	44
6.1.8. Tela de criação do ponto de doação.....	44
6.1.9. Tela de visualização do estoque.....	45
6.1.10. Tela de controle do estoque.....	46
6.1.11. Tela de configurações do perfil do gerente.....	47
6.1.12. Tela de história do website.....	48
6.1.13. Tela de informações de contato.....	48
7. CONSIDERAÇÕES FINAIS.....	50
REFERÊNCIAS.....	51

1. INTRODUÇÃO

Atualmente, no Brasil, de acordo com uma pesquisa realizada em 2022 pela Rede Brasileira de Pesquisa em Soberania e Segurança Alimentar e Nutricional (Rede PENSSAN), aproximadamente 33,1 milhões de pessoas passam fome. Além disso, ao todo cerca de 125,2 milhões estão situadas em algum grau de insegurança alimentar. Em comparação com o inquérito realizado no fim de 2020, existiam 19,1 milhões de pessoas convivendo com a fome, ou seja, ao longo desses 2 anos ocorreu um aumento de 57,40% de pessoas em estado severo de fome.

Dessa forma, é inegável a necessidade de implementar medidas que busquem combater o avanço da insegurança alimentar que atinge mais da metade da população brasileira. Em diversas regiões existem órgãos públicos e organizações não governamentais (ONGs) que arrecadam alimentos e roupas para doar aos mais necessitados. Entretanto, em decorrência da falta de informação, diversas pessoas não têm conhecimento desses centros de doação e ajuda.

Nesse contexto, o DoeAqui busca auxiliar na localização e gerenciamento do centro de doação mais próximo do usuário. Dessa forma, por meio de uma simples busca, qualquer pessoa pode encontrar e se dirigir a um centro de doação seja para contribuir, ou para receber auxílio.

Por último, espera-se que com a implementação do *website* DoeAqui, o processo de arrecadação seja otimizado, assim como, o processo de distribuição e gerenciamento. Facilitando para o público carente e os doadores encontrarem um centro de distribuição próximo.

Este trabalho está organizado em 7 partes. Depois da introdução, a segunda parte apresenta a justifica ao redor do trabalho. Na parte 3, estão localizados os objetivos gerais e objetivos específicos, definidos ao redor de 3 premissas principais necessárias para concluir o projeto. Na parte 4, é abordado a fundamentação teórica do projeto, ou seja, são apresentados as pesquisas e referências bibliográficas. Na parte 5, a expõe a metodologia utilizada e a descrição dos passos a passos adotados na construção do sistema. Na parte 6, são apresentados os resultados obtidos. Por último, na parte 7 está a conclusão do trabalho.

2. JUSTIFICATIVA

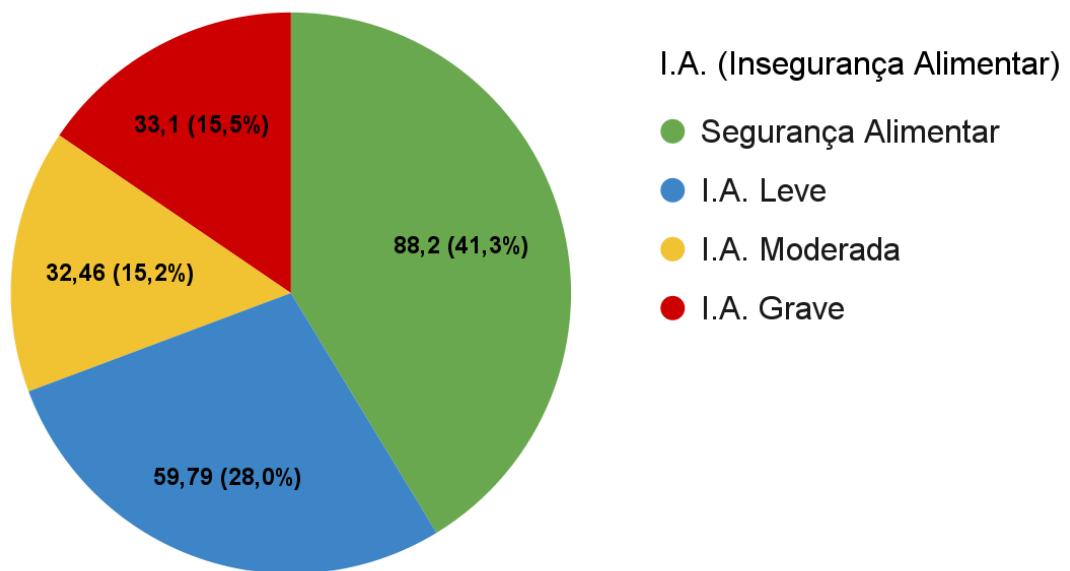
A meta do DoeAqui é auxiliar no combate à fome que se alastrou no Brasil, em decorrência da pandemia que iniciou-se em 2020. Por causa disso, é preciso desenvolver um software que ajude as famílias que estão em situação de risco alimentar e financeiro. O programa proposto neste trabalho pretende auxiliar os centros de arrecadação tanto no recolhimento de alimentos, roupas e itens higiênicos, quanto na distribuição das doações.

Além disso, torna mais fácil para o público carente localizar um centro de distribuição próximo do usuário, visto que, o sistema utilizará recursos de geolocalização para ajudar na locomoção até o destino. Tendo em vista isso, o DoeAqui fornece uma forma de localização mais acessível, ao invés de apenas citar o endereço do site e deixar a cargo do usuário filtrar o ponto mais próximo. Além do mais, os centros podem listar os itens que estão em falta. Desse jeito, os voluntários saberão exatamente qual tipo de produto o centro está necessitando, já que podem ocorrer casos de excesso de alguns produtos e falta de outros.

Espera-se que a medida que o *website* se popularizar, torne-se cada vez mais fácil contribuir para o combate à fome e à miséria , visto que, é um mal que ainda assola a população brasileira, dado que, apenas 41,3% da população brasileira está em estado de segurança alimentar (REDE PENSSAN, 2022). A figura 1 mostra os níveis de segurança alimentar no Brasil. Em 2022, 33, 1 milhões de pessoas estão em I.A. (Insegurança Alimentar) grave, 32,46 milhões em I.A. moderada e 59,79 milhões em I.A. leve, esses dados demonstram a situação alarmante de mais da metade da população brasileira. O DoeAqui beneficiará diversas pessoas que estão em situação de risco, contribuindo para uma melhor qualidade de vida desses indivíduos, sobretudo no aspecto da segurança alimentar.

Figura 1: Gráfico sobre a segurança alimentar no Brasil em milhões de pessoas.

Segurança Alimentar no Brasil em milhões de pessoas



Fonte: (REDE PENSSAN, 2022).

3. OBJETIVOS

3.1. OBJETIVO GERAL

Desenvolver um *website* para localizar e cadastrar pontos de arrecadação de alimentos, roupas e itens higiênicos. Por meio de um recurso de geolocalização, o site determinará o ponto mais próximo do usuário. Desse modo, os doadores e necessitados podem localizar o centro mais próximo da sua localização.

3.2. OBJETIVOS ESPECÍFICOS

- a) Desenvolver o cadastramento dos centros de arrecadação.
- b) Criar rotinas para o gerente do ponto de arrecadação administrar os itens a serem doados.
- c) Implementar o uso do Google Maps API para rastrear e filtrar o ponto de coleta mais próximo.

4. FUNDAMENTAÇÃO TEÓRICA

Este capítulo contém a fundamentação teórica deste projeto, esclarecendo os conceitos fundamentais para possibilitar sua compreensão plena.

4.1. SISTEMAS WEB

Até meados da década de 1990, a internet era utilizada majoritariamente por pesquisadores e acadêmicos para transferir arquivos entre servidores, receber notícias e correios eletrônicos. Entretanto, ela era desconhecida pela maior parte do público em geral. Somente em 1990, com a *World Wide Web* proposta por Tim Berners-Lee que a internet chamou a atenção das pessoas.

Segundo Kurose e Ross (2013), a *Web* é uma aplicação cliente-servidor que proporciona aos usuários receberem documentos dos servidores por meio de requisições, dessa forma, tornando possível o acesso remoto de arquivos hospedados em outras máquinas. Em outras palavras, as aplicações *web* são soluções que buscam simplificar e agilizar processos, tanto de empresas, quanto de pessoas. Deste modo, a complexidade aumenta conforme são adicionadas múltiplas funcionalidades.

4.1.1. BACK-END

Abstraindo o conceito de, *back-end*, assim como o nome sugere, significa a parte por trás da aplicação. Desse jeito, pode-se pensar que para realizar login nas redes sociais seus dados precisam estar armazenados em um banco de dados. Ele é a ponte entre as informações que vem do navegador *web*, por exemplo Google Chrome, rumo ao banco de dados onde as informações estão armazenadas e vice-versa. Ademais, aplica as regras de negócios e validações, propiciando ambiente seguro dos dados. As principais linguagens do *back-end* são Javascript, PHP, Java, ASP e Python (SOUTO, 2019).

Além do mais, neste projeto foi utilizado Node.Js, no *back-end*. O Node é capaz de interpretar um código *JavaScript*, igual ao que o navegador faz. Sendo assim, quando o navegador recebe um comando em *JavaScript*, ele o interpreta e depois executa as instruções fornecidas (PESSOA, 2022). Deste modo, torna-se possível integrar a outros sistemas, como por exemplo, o banco de dados firestore utilizado para armazenar arquivos no formato JSON. O JSON (*JavaScript Object Notation*) é um formato que armazena informações estruturadas e

é principalmente usado para transferir dados entre um servidor e um cliente (LONGEN, 2021). Dessa maneira, quando utilizado no JavaScript, é possível interpretar os dados no formato de um objeto. Por fim, essa parte relacionada aos servidores é considerada o server-side.

4.1.2. FRONT-END

O *Front-end* pode ser definido como a parte visual e interativa de um site, ou seja, o *client-side*. Em vista disso, os desenvolvedores *Front* são responsáveis por programar a interface gráfica, geralmente, utilizando tecnologias, como por exemplo, HTML (*HyperText Markup Language*), CSS (*Cascading Style Sheets*) e Javascript, permitindo usuários interagirem com a página *web* e torná-la dinâmica. Segundo Flatschart (2011), esse agrupamento em camadas independentes proporciona flexibilidade e modularidade na sucessão do trabalho *web*.

O HTML permite criar a estrutura básica de um site na internet. Embora ele não seja propriamente uma linguagem de programação, usando marcações de códigos chamados “Tags”, ele possibilita que informações sejam organizadas e formatadas para dar estrutura e forma à página (RESENDE, 2021). Alinhado a isso, o CSS é usado para estilizar elementos escritos no arquivo HTML. O CSS separa o conteúdo da representação visual do site (RUCKERT, 2022). Por último, o JavaScript é uma linguagem de programação de alto nível voltada para o desenvolvimento *web*, criada originalmente para funcionar do lado do usuário, ou seja, nos navegadores (MELO, 2021).

Por último, integrado ao *client-side* estão as APIs (*Application Programming Interface*). Uma API é uma interface que permite a interação entre os softwares facilitando sua integração (FLATSCHART, 2011). Sendo assim, diferentes aplicativos podem se comunicar entre si, por exemplo, ao acessar um site de hospedagem que possui um mapa do Google Maps indicando a localização do hotel. Mesmo que o site do hotel não seja produzido pela Google ele está ligado a ela por meio de uma API.

4.2. TESTES E USABILIDADE

É possível definir dois principais motivos para testar um software. O primeiro motivo busca apresentar ao desenvolvedor e ao cliente que o sistema atende aos requisitos acordados. O segundo motivo busca encontrar inconsistências ou comportamentos indesejados, causados

por defeitos (*bugs*), dessa forma, o teste de software busca encontrá-los para eliminar os comportamentos, falhas de sistemas indesejadas ou interações incorretas (SOMMERVILLE, 2019).

Aliás, eles podem ser divididos em diversas categorias, como por exemplo, teste de caixa branca, teste de caixa preta e teste de usabilidade. O teste de caixa branca recebe esse nome pois o testador possui acesso ao código fonte e estrutura interna do sistema, dessa maneira, o profissional analisa o caminho do fluxo dos dados, validando a lógica da aplicação, porém esse teste exige mais conhecimento técnico, ocasionando um custo maior de realização (INTHURN, 2001). O teste de caixa preta é oposto ao caixa-branca, o testador não tem acesso ao código, o objetivo é validar as entradas e saídas do sistema, portanto o testador está preocupado em garantir a coerência das funcionalidades do sistema (BARTIE, 2002).

Por fim, o teste de usabilidade busca verificar a experiência do usuário. Uma ótima interface deve ser esteticamente agradável e permitir que o usuário a utilize facilmente. Em outras palavras, faça todas as tarefas que desejar de maneira simples e direta. Segundo Jakob Nielsen (2000), a funcionalidade pode ser dividida em 5 fatores: facilidade de aprender (*learnability*), facilidade de lembrar (*memorability*), eficiência (*efficiency*), segurança (*safety*) e satisfação (*satisfaction*).

A facilidade de aprender está relacionada com o tempo e esforço necessários para atingir certo nível de compreensão da aplicação. A facilidade de recordação está ligada ao esforço para lembrar como utilizar o sistema. A eficiência está relacionada ao tempo de realização de uma tarefa com o uso do sistema. A segurança de uso tem a ver com a proteção do usuário contra possíveis erros. Por último, a satisfação do usuário, desejavelmente deve ser alta, pode ser medida através de formulários de opinião.

Nesse contexto, o uso dos testes são indispensáveis para avaliar se todos os requisitos tanto de usabilidade, regras de negócio e lógica estão sendo implementados corretamente no sistema.

4.3. ENGENHARIA DE REQUISITOS

Os requisitos de um software representam as características dos serviços que ele deve prestar e as restrições de sua atividade. Além disso, eles são um reflexo das necessidades dos clientes que utilizam a aplicação. O método de exploração, análise, documentação e exposição dos serviços prestados e restrições é denominado engenharia de requisitos (SOMMERVILLE, 2019).

Os requisitos funcionais representam os problemas e necessidades que precisam ser atendidos e resolvidos pelo sistema. Os requisitos não funcionais representam a maneira como o sistema realizará suas atividades, apesar de não estar diretamente ligado às funcionalidades do sistema (SOMMERVILLE, 2019).

4.4. ARQUITETURA DE SOFTWARE

A arquitetura refere-se a um conceito abstrato de como o código tem que ser organizado e a projeção estrutural comum do sistema. Dessa forma, o resultado obtido desse processo é um modelo que detalha a maneira como o software está disposto em um agrupamento de componentes que se comunicam entre si mesmo (SOMMERVILLE, 2019).

Os principais benefícios de implementação de uma arquitetura de software são: performance, escalabilidade e flexibilidade.

A performance está relacionada ao desempenho do sistema, sendo assim, sistemas mal estruturados tornam a manutenção mais trabalhosa. Uma arquitetura bem definida facilita o cumprimento das demandas e facilita o controle do volume dos dados. Outro ponto importante é a escalabilidade. Um projeto escalável facilita os futuros processos, evita atrasos de entrega e erros posteriores. Finalmente existe a flexibilidade, já que toda empresa tem seus requisitos únicos e necessita de um sistema que adeque-se às suas necessidades.

Além de tudo, outro fator crucial é a escolha do padrão arquitetural. Eles são soluções práticas sugeridas que passaram por diversos processos de estudo, testes e aprovações por apresentarem resultados bem-sucedidos em sistemas implementados anteriormente (SOMMERVILLE, 2019).

4.5. GEOLOCALIZAÇÃO

A geolocalização é o processo de localizar algo ou alguém no planeta através de coordenadas geográficas. São ainda sistemas de referência que utilizam coordenadas angulares, sendo elas a latitude (norte e sul) e longitude (leste e oeste). A latitude representa a distância de qualquer ponto da superfície terrestre em relação a linha do equador. Ao mesmo tempo, a longitude representa a distância entre qualquer ponto da superfície terrestre em relação ao meridiano de Greenwich (OLIVEIRA, 2020). Este meridiano é uma linha imaginária que divide a superfície entre os hemisférios oeste e leste e representa o marco zero para a contagem universal das longitudes e também das horas (GUITARRARA, 2022).

Para este projeto, utilizaremos serviços de geolocalização disponíveis por uma API do Google Maps JavaScript, que consistem em um serviço público disponibilizado pela Google LLC. Ela possui uma versão *open source*, ou seja, seu código fonte é disponibilizado gratuitamente, desde que o usuário final não seja cobrado, ela também possui uma versão paga. Através desta API, é possível incorporar um mapa do Google na página *web*, possibilitando a manipulação do mapa, marcação de pontos com ícones personalizados, desenhar formas geométricas, trajetos de navegação e entre outras coisas (GOOGLE, 2022).

4.6. BIBLIOTECAS E FRAMEWORKS

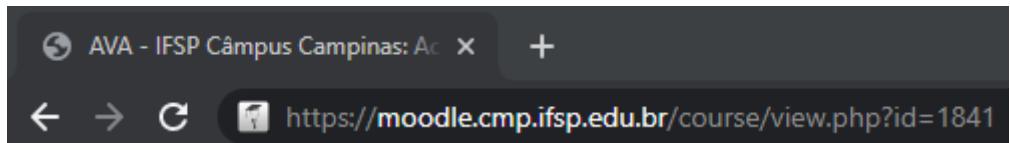
As bibliotecas são coleções de códigos prontos criados para resolver determinados tipos de problemas. Enquanto isso, o conceito de framework é ainda mais abstrato, ele busca agir como um suporte que representa um conjunto de funcionalidades prontas com um fluxo ou estrutura a ser seguida. Dessa forma, enquanto na biblioteca o desenvolvedor decide quando e onde utilizar as funções, no framework o desenvolvedor segue uma estrutura definida (ZANETTE, 2017).

4.7. ROTAS DO SISTEMA

As rotas do sistema representam padrões de URL (*Uniform Resource Locator*) mapeadas para um controlador da ação, ou seja, dependendo do caminho da URL um determinado controlador será requisitado. A URL é um endereço eletrônico que possibilita acessar um recurso tecnológico na internet, por exemplo, um site (BERNERS-LEE; MASINTER; MCCAHILL, 1994).

A figura 2 exemplifica o formato de uma URL. Ela é dividida em protocolo, *host*, *path* e parâmetros. As URLs iniciam com o protocolo que o servidor utiliza, por exemplo, o HTTPS (*Hypertext Transfer Protocol Secure*). O *host* refere-se ao nome que identifica a máquina na internet (moodle.cmp.ifsp.edu.br). O *path* demonstra a forma de localizar o recurso dentro do servidor (/course/view.php). Os parâmetros são os valores anexados à URL (id=1841).

Figura 2: Exemplo de URL.



Fonte: Autoria própria (2022).

4.8. TEMPLATE ENGINE

Uma *template engine* é uma ferramenta usada para ajudar na criação de páginas HTML e transformar o envio e exibição de dados para estas páginas um processo mais simples e organizado (ANDRADE, 2020). O *Nunjuncks* (NJK) é uma *template engine*, ela permite a renderização de uma página HTML com um código JavaScript integrado, por meio de um arquivo njk (COVO, 2019).

4.9. FIREBASE FIRESTORE E FIREBASE AUTHENTICATION

O *Firebase* é um conjunto de serviços de hospedagem para simplificar o desenvolvimento de uma aplicação, seja ela *web* ou *mobile*. Com esta plataforma, o processo do back-end da aplicação é facilitado e torna possível focar apenas no front-end, economizando tempo e recursos (ANDRADE, 2021).

O *Firebase Firestore* é um banco de dados em nuvem, não relacional e orientado a documentos que pode ser usado para armazenar e consultar dados como objetos no formato JSON. O Cloud Firestore facilita o desenvolvimento flexível de dispositivos móveis, *web* e servidor. Também pode ser usado para sincronizar dados em aplicativos em tempo real. Ele está integrado ao Google Cloud e Firebase (CLARK, 2020).

O *Firebase Authentication* providencia serviços *back-end*, *SDKs* (*Software Development Kit*), ou seja, um conjunto de ferramentas de desenvolvimento que possibilita a produção de aplicativos, e bibliotecas preparadas para autenticar usuários no sistema (FIREBASE, 2022).

4.10. FERRAMENTAS E TECNOLOGIAS UTILIZADAS

A IDE (*Integrated Development Environment*), utilizada foi o VS Code (Visual Studio Code). Lançado pela *Microsoft*, é um editor de código destinado ao desenvolvimento de aplicações *web* (DIONISIO, 2016).

Para a criação das telas foi utilizado o Figma, um editor gráfico de vetor baseado na *web* utilizado principalmente para trabalhos de prototipagem de projetos de design (HARADA, 2022).

Além do mais, para realizar o controle do versionamento dos arquivos foi adotado o *Git*, um software livre muito utilizado no desenvolvimento de aplicações onde diversas pessoas estão contribuindo simultaneamente, podendo criar e editar arquivos (GUEDES, 2019). Integrado ao git foi utilizado o GitHub, uma plataforma para hospedar e armazenar projetos. Ele pode ser comparado a uma rede social, só que ao invés de fotos, os desenvolvedores podem adicionar seus códigos e disponibilizá-los para outras pessoas verem e usufruírem (GUEDES, 2019).

5. METODOLOGIA

Para o desenvolvimento do sistema *Web* foram necessárias as etapas de análise de requisitos, criação do layout, arquitetura e rotas do sistema, banco de dados e autenticação, localização do ponto mais próximo, teste e hospedagem da aplicação. A explicação aprofundada está descrita em suas respectivas subseções.

5.1. ANÁLISE DE REQUISITOS

A etapa de análise de requisitos aconteceu por meio de um levantamento realizado na *web* em busca de sites semelhantes ao objetivo do DoeAqui. Ao longo da coleta de dados, foram analisadas semelhanças e diferenças entre as páginas, como por exemplo, a descrição do objetivo, responsividade e endereço do ponto de arrecadação. Por conseguinte, os dados analisados foram reunidos, como pode ser observado na tabela 1. Deste modo, foi possível também levantar os requisitos funcionais e não funcionais do sistema. Os sites analisados foram: CEASA, Casa da Sopa e a FEAC.

O CEASA Campinas, é uma empresa criada pela prefeitura de Campinas que fornece apoio a programas sociais (CEASA, 2022). A Casa da Sopa é uma entidade benéfica que ajuda famílias carentes na região do Ouro Verde (CASA DA SOPA, 2022). A Fundação FEAC é uma organização independente que atua em Campinas investindo em ações de educação, assistência social e promoção humana com foco nas regiões e nas populações mais vulneráveis, especialmente crianças e adolescentes (FEAC, 2022).

Tabela 1: Principais características de sites de arrecadação.

Características	DoeAqui	CeasaCampinas	CasaDaSopa	Feac
Descrição	x	x	x	x
Responsividade	x	x	x	
Localização	x	x	x	x
Suprimentos	x			x
Mapa dos pontos	x			
Cadastro do ponto	x			
Buscar mais próximo	x			

Fonte: Autoria própria (2022).

O cadastramento das instituições é essencial para expandir a área de aplicação do sistema, também é importante garantir que todas as informações inseridas sejam válidas. A responsividade está relacionada à capacidade do sistema adaptar-se a diferentes dispositivos, facilitando a visualização dos pontos e a interação com o sistema. A localização está ligada à forma de encontrar o centro de doações. O DoeAqui exibe um ícone marcando a localização do ponto.

O mapa dos pontos é o principal diferencial do site, visto que, é muito mais fácil distinguir o ponto mais próximo olhando diretamente para o mapa, unido a isso o botão de localizar já fornece o centro cadastrado mais próximo do usuário. Ademais, um diferencial apresentado por um dos sites foi a descrição dos suprimentos, ou seja, os itens que estão em falta ou em abundância no local. Por último, é importante descrever o objetivo do centro, porque não só identifica sua funcionalidade, sendo ela doar roupa ou comida, mas também demonstra o sentimento de solidariedade empregado.

5.1.1. REQUISITOS FUNCIONAIS

A lista abaixo representa os principais requisitos funcionais do sistema:

- Localizar centros de doação mais próximos do usuário.
- Filtrar centros de doação por categorias, seja o lugar focado em doar roupa, alimento ou produtos de higiene.
- Cadastrar centros de doação, validando o e-mail e localização.
- Rotina para gerenciar o nível do estoque dos produtos a serem doados.
- Guia no *website* para contatar em caso de dúvida, reclamação ou sugestão de melhorias.

5.1.2. REQUISITOS NÃO FUNCIONAIS

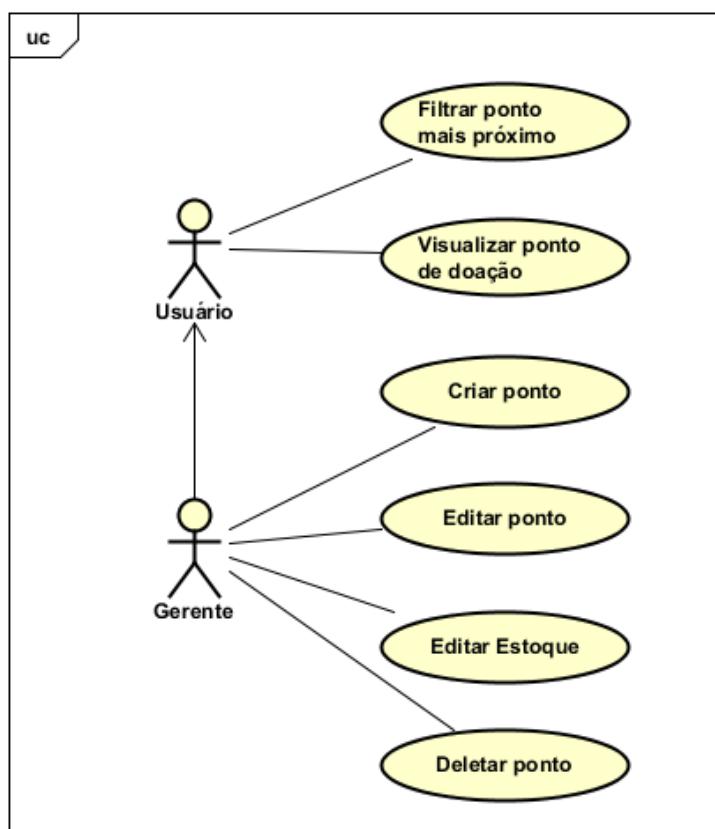
A lista abaixo representa os principais requisitos não funcionais do sistema:

- Compatibilidade e ajuste de tela de acordo com o tipo de dispositivo.
- Confiabilidade de precisão no momento de rastrear a localização do cliente até o centro.
- Rápida velocidade de execução ao utilizar o sistema.
- Proteção das informações do usuário através da criptografia implementado pelo *Firebase Authentication*.

5.2. ESTUDO DE CASO DE USO

O estudo do diagrama de caso de uso (Figura 3) facilita a compreensão de quais são as funcionalidades da aplicação e seus níveis de acesso. O usuário comum pode apenas filtrar o ponto de doação mais próximo e visualizar as informações de cada centro de coleta. Enquanto isso, o gerente além de poder realizar as mesmas ações do usuário comum ele também pode criar um novo ponto, editar os pontos cadastrados, editar os estoques e deletar os pontos de doações.

Figura 3 - Diagrama de Caso de Uso



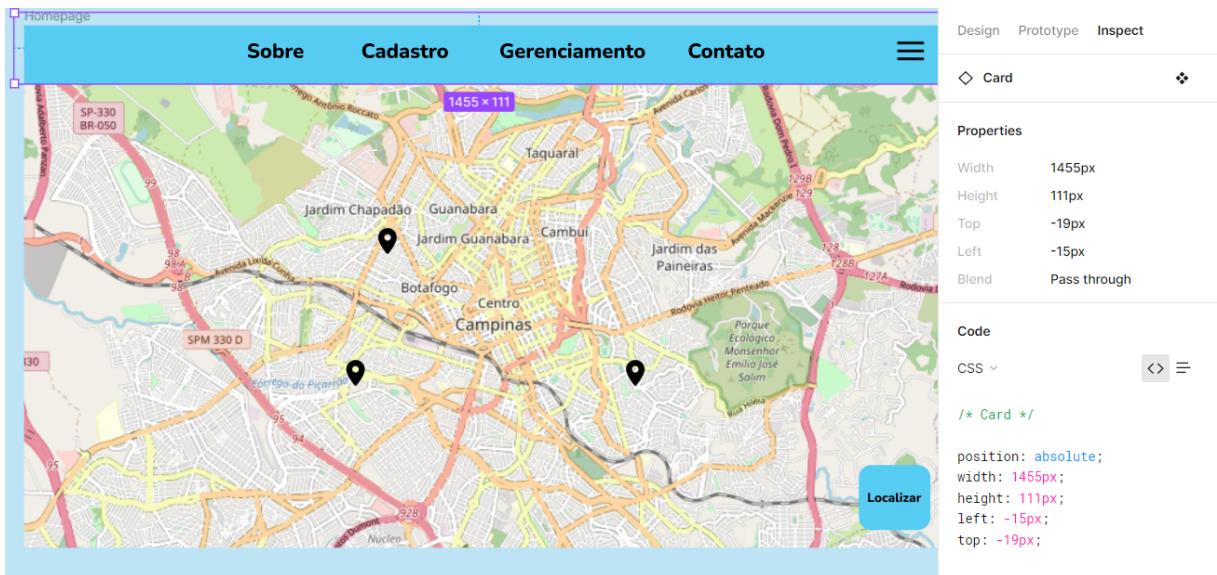
Fonte: Autoria própria (2022).

5.3. CRIAÇÃO DO LAYOUT

O *design* do *layout* foi construído utilizando o software Figma, buscando atender os cinco atributos de usabilidade de Jakob Nielsen, portanto facilitando o uso da ferramenta para os clientes. Focando na *learnability*, visto que, o usuário normalmente não têm conhecimento da interface, em sua primeira experiência, ou seja, ele vai descobrindo suas funcionalidades

durante o uso da aplicação. Além do mais, o Figma também fornece a estilização da página em código no formato de CSS (Figura 4).

Figura 4: Prototipação de telas com o Figma.

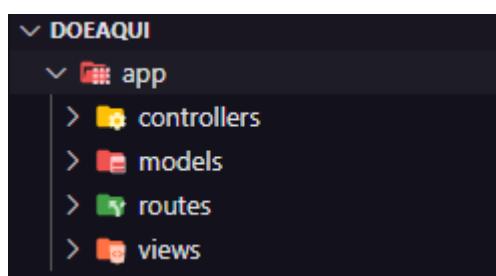


Fonte: Autoria Própria

5.4. ARQUITETURA E ROTAS DO SISTEMA

O padrão arquitetural adotado foi o *Model-View-Controller* (MVC). Ele é responsável por otimizar as requisições e respostas realizadas pelo sistema. A arquitetura é dividida em três componentes: *Model*, *View* e *Controller*. Dessa forma, facilita a organização, divisão de tarefas, segurança e eficiência do sistema (Figura 5). Enquanto isso, as *routes* gerenciam todos os caminhos acessíveis do sistema.

Figura 5: Organização estrutural do sistema



Fonte: Autoria Própria

5.4.1. MODEL

O *model* é responsável por gerenciar e controlar o comportamento dos dados, através das regras de negócio, funções e lógica. A figura 6 exemplifica o comportamento do *model* durante a criação de um centro. A função recebe os dados filtrados pelo *controller* e adiciona o documento na coleção *centers*, caso a operação seja bem sucedida retorna um identificador único e caso ocorra um erro retorna uma mensagem informando erro.

Figura 6: Exemplo de código do *model* da criação de um centro



```

point.js  ×

...
1  const { db } = require("../config/firebase");
2  const { addDoc, collection, deleteDoc, doc, getDoc, getDocs, query, where, updateDoc } = require
   ("firebase/firestore");
3
4  const createCenter = async (dataCenter) => {
5      try {
6          const center = await addDoc(collection(db, "centers"), dataCenter);
7          return center.id;
8      } catch (err) {
9          console.error("Erro salvando o centro: ", err);
10     }
11 }
12

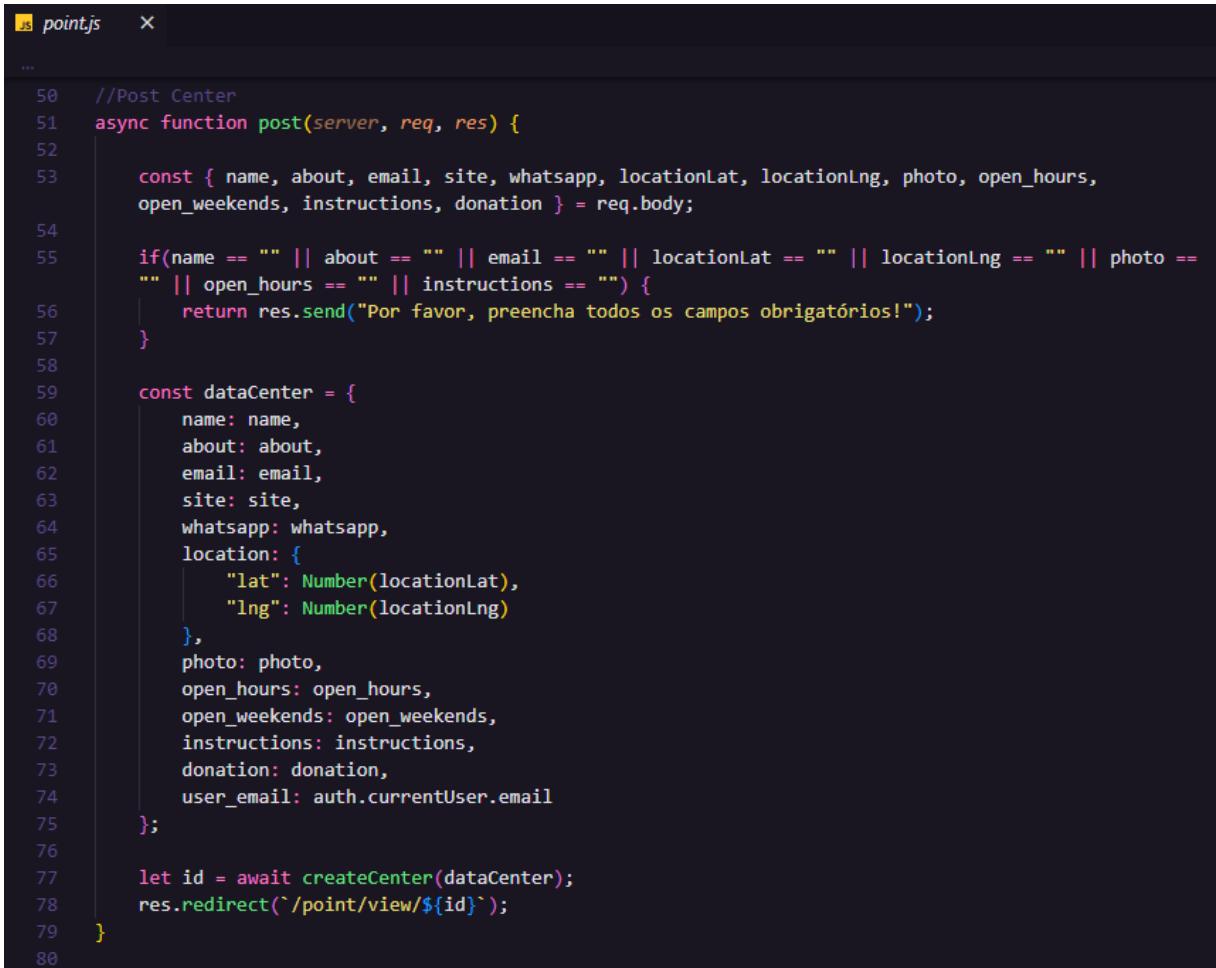
```

Fonte: Autoria Própria

5.4.2. CONTROLLER

O controller é responsável por intermediar as requisições da *View* e as respostas do Model. A figura 7 exemplifica o comportamento do *controller* durante a criação de um centro. Os campos são validados: caso haja um campo vazio, o sistema retorna uma mensagem solicitando o preenchimento de todos os campos. Após isso, os dados são incluídos em um objeto denominado *dataCenter* e logo em seguida a função *createCenter* do *model* é chamada. Por fim, caso a operação seja bem sucedida um identificador único é retornado e o gerente é redirecionado para o novo centro criado.

Figura 7: Exemplo de código do *controller* da criação de um centro



```

js point.js  X

...
50  //Post Center
51  async function post(server, req, res) {
52
53      const { name, about, email, site, whatsapp, locationLat, locationLng, photo, open_hours,
54          open_weekends, instructions, donation } = req.body;
55
56      if(name == "" || about == "" || email == "" || locationLat == "" || locationLng == "" || photo == ""
57          "" || open_hours == "" || instructions == "") {
58          return res.send("Por favor, preencha todos os campos obrigatórios!");
59      }
60
61      const dataCenter = {
62          name: name,
63          about: about,
64          email: email,
65          site: site,
66          whatsapp: whatsapp,
67          location: {
68              "lat": Number(locationLat),
69              "lng": Number(locationLng)
70          },
71          photo: photo,
72          open_hours: open_hours,
73          open_weekends: open_weekends,
74          instructions: instructions,
75          donation: donation,
76          user_email: auth.currentUser.email
77      };
78
79      let id = await createCenter(dataCenter);
80      res.redirect(`#/point/view/${id}`);
}

```

Fonte: Autoria Própria

5.4.3. VIEW

A View é responsável por apresentar visualmente as respostas para o usuário do sistema. Quando um gerente acessa a rota de criar um novo centro de doação, o *controller* responde renderizando o arquivo *create.njk* da *view* (Figura 8).

Figura 8: Exemplo de código da *view* da criação de um centro

```

Nj create.njk ×
nenhum elemento
1  {% extends "layout.njk" %} 
2
3  {% block head %}
4    <title>DoeAqui - Criação</title>
5  {% endblock head %}
6
7  {% block content %}
8    <link rel="stylesheet" href="/styles/form.css">
9    <link rel="stylesheet" href="/styles/buttons.css">
10   <link rel="stylesheet" href="/styles/view.css">
11
12  <a href="javascript:void(0)" onClick="history.go(-1); return false;" class="btn-come-back">Voltar</a>
13
14  <form method="POST" action="/point">
15    {% include 'point/fields.njk' %}
16
17    <button type="submit" class="primary-button">Salvar</button>
18  </form>
19
20  <!-- Google Maps API -->
21  <script async defer src="https://maps.googleapis.com/maps/api/js? key=AIzaSyA2rNZNpz1TbudC8qHoXsNN8i3LGbYWa0&callback=initMap"></script>
22
23  <!-- Lib jquery -->
24  <script src="https://ajax.googleapis.com/ajax/libs/jquery/2.1.1/jquery.min.js"></script>
25  <script type="text/javascript" src="https://cdnjs.cloudflare.com/ajax/libs/jquery.mask/1.14.0/jquery.mask.js"></script>
26
27  <script src="/scripts/mapCreate.js"></script>
28  {% endblock content %}

```

Fonte: Autoria Própria

5.4.4. ROUTES

O *routes* é responsável por gerenciar os caminhos do sistema, fornecendo uma resposta correspondente ao caminho solicitado. O routes implementado utiliza o framework Express.js, para auxiliar no funcionamento do servidor. Quando um usuário acessa o caminho raiz do sistema o *routes* chama a função *home* do *controller* para intermediar as requisições (Figura 9).

Figura 9: Exemplo de código das *routes* do sistema

```

1  const { home , centers} = require('../controllers/home');
2  const { about, contact } = require('../controllers/info');
3
4  module.exports = {
5      home: (server) => {
6          server.get("/", (req, res) => {
7              home(server, req, res);
8          });
9      },
10     centers: (server) => {
11         server.get("/centers", (req, res) => {
12             centers(server, req, res);
13         });
14     },
15     about: (server) => {
16         server.get("/about", (req, res) => {
17             about(server, req, res)
18         });
19     },

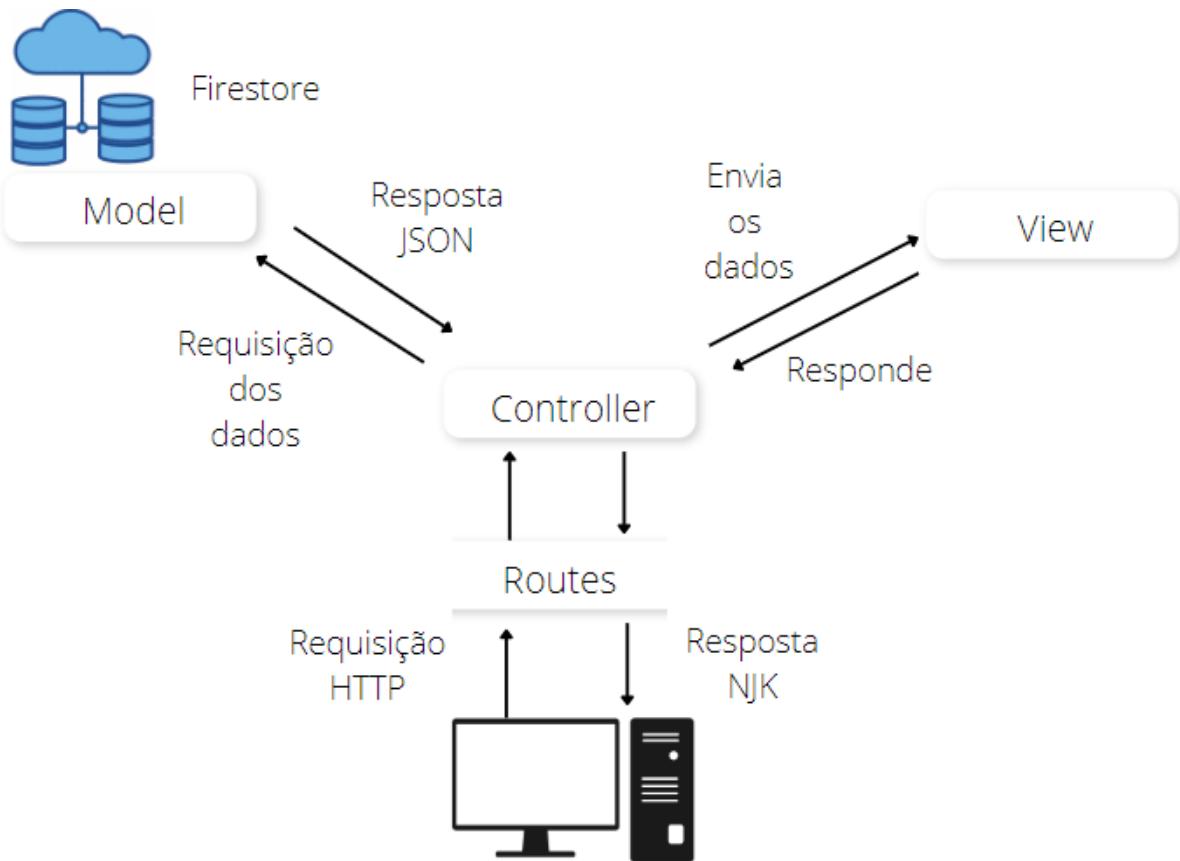
```

Fonte: Autoria Própria

5.4.5. VISÃO GERAL

Ao acessar o sistema, o usuário realiza uma requisição HTTP, administrada pelo *routes* e repassada para o devido *controller*, que realiza uma requisição dos dados para model conectado ao banco de dados *firestore*, a resposta é um arquivo JSON, repassada para a *view*, pelo *controller*, a resposta é enviada para o usuário em formato um nunjucks (Figura 10).

Figura 10: Visão geral do MVC

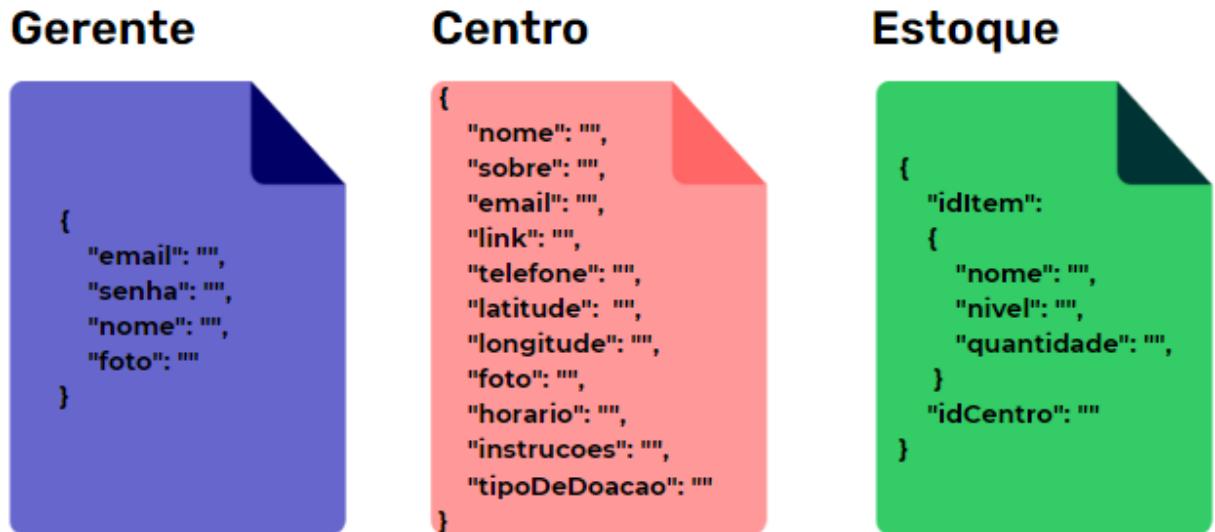


Fonte: Autoria Própria

5.5. BANCO DE DADOS E AUTENTICAÇÃO

O banco de dados e a autenticação são compostos por 3 documentos: gerente, centro e estoque, conforme ilustrado na figura 11. Além do mais, os dados quando chamados podem ser utilizados diretamente pelo sistema como objetos.

Figura 11 - Documentos do banco de dados



Fonte: Autoria própria (2022).

5.5.1. GERENTE

O gerente é controlado pelo *Firebase Authentication*, quando um usuário cria uma conta o e-mail e a senha são armazenados automaticamente pelo *firebase authentication*: após ter criado a conta o gerente pode adicionar um nome e foto de perfil na página de configurações do perfil (Figura 29). Vale ressaltar que, o administrador não tem acesso a nenhum dado interno dos gerentes, apenas ao e-mail (Figura 12).

Figura 12 - Visão do administrador do Firebase *Authentication*

The screenshot shows the Firebase Authentication interface under the 'Users' tab. At the top, there are tabs for 'Users', 'Sign-in method', 'Templates', 'Usage', and 'Settings'. Below the tabs is a search bar with placeholder text 'Pesquise por endereço de e-mail, número de telefone ou UID do usuário'. To the right of the search bar is a blue button labeled 'Adicionar usuário'. Further to the right are icons for refresh and more options. The main area displays a table of users:

Identificador	Provedores	Data de criação	Último login	UID do usuário
	✉	18 de set. d...	19 de out. d...	KML37QejdcWF1pYocx6UoxVKI5...
teste01@gmail.com	✉	10 de set. d...	18 de set. d...	rs2CFJBj5wZy084x43wKpyx6m6E3
teste@gmail.com	✉	10 de set. d...	19 de out. d...	nLOG4HE9DMNbSpbM0cChPB9Tq...

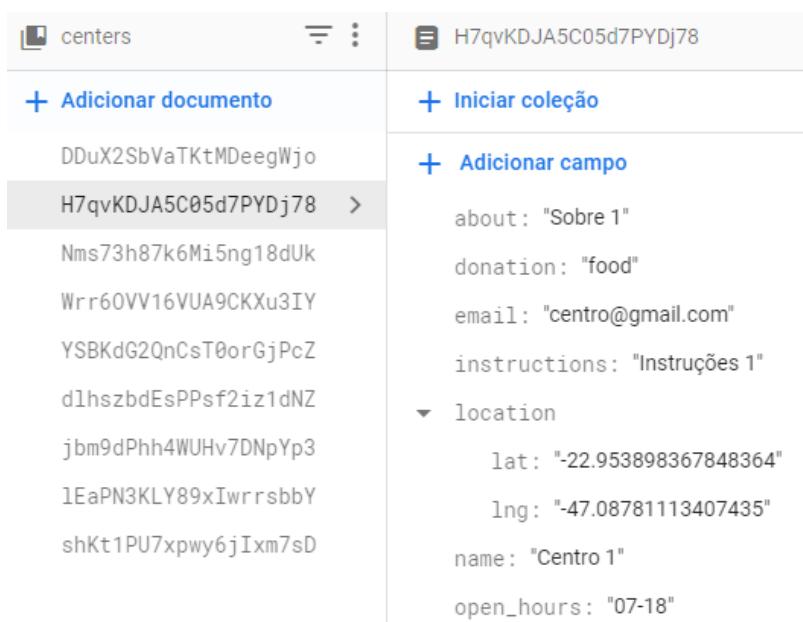
At the bottom of the table, there are pagination controls: 'Linhas por página: 50', '1 - 3 of 3', and navigation arrows.

Fonte: Autoria própria (2022).

5.5.2. CENTRO E ESTOQUE

Conforme demonstrado pelo documento de banco de dados (Figura 11), os dados armazenados pelo centro são nome, história do lugar, e-mail de contato, telefone, link (caso o gerente deseje adicionar um *website* ou endereço de alguma página web), latitude e longitude, foto do centro, horário de funcionamento, instruções e o tipo de doação que o local recebe. Quando um gerente cria um centro, o *Firestore* gera uma chave de identificação única do documento (Figura 13). Enquanto isso, cada item do estoque possui uma chave de identificação única e dentro desse objeto são armazenados o nome, nível (seja um item que está com baixo, médio ou alto estoque) e quantidade. Por fim, cada estoque possui uma propriedade chamada *idCentro* que identifica o centro ao qual o estoque está associado (Figura 14).

Figura 13 - Visão do administrador do documento dos centros



The screenshot shows the Firestore console interface. On the left, there's a sidebar with a 'centers' collection and several document IDs: DDuX2SbVaTKtMDeegWjo, H7qvKDJA5C05d7PYDj78, Nms73h87k6Mi5ng18dUk, Wrr60VV16VUA9CKXu3IY, YSBKdG2QnCsT0orGjPcZ, dlhszbdeEsPPsf2iz1dNZ, jbm9dPhh4WUHv7DNpYp3, 1EaPN3KLY89xIwrrsbbY, and shKt1PU7xpwy6jIxm7sD. The document 'H7qvKDJA5C05d7PYDj78' is selected and expanded. On the right, the document details are shown:

```

about: "Sobre 1"
donation: "food"
email: "centro@gmail.com"
instructions: "Instruções 1"
location:
  lat: "-22.953898367848364"
  lng: "-47.08781113407435"
  name: "Centro 1"
  open_hours: "07-18"

```

Fonte: Autoria própria (2022).

Figura 14 - Visão do administrador do documento dos estoques

```

{
  "data": {
    "arroz5kg": {
      "amount": "30",
      "level": "1",
      "name": "Arroz 5kg"
    },
    "farinhadetrigo": {
      "amount": "70",
      "level": "2"
    },
    "feijao1kg": {
      "amount": "70",
      "level": "2"
    },
    "leite1l": {
      "amount": "12",
      "level": "0"
    },
    "oleodesoja": {
      "amount": "4",
      "level": "0"
    }
  },
  "idCenter": "jbm9dPhh4WUHv7DNpYp3"
}

```

Fonte: Autoria própria (2022).

5.6. LOCALIZAÇÃO DO PONTO MAIS PRÓXIMO

A filtragem do ponto mais próximo do usuário é implementada pela função *findClosestN* (Figura 15). Inicialmente é declarado um array vazio denominado *closest*. Após isso, é chamado um *for* para percorrer todos os pontos marcados no mapa, dentro do *for* a variável *marker* recebe uma propriedade *distance*, contendo o resultado entre a distância da localização do usuário e o ponto de doação. Cada ponto é eliminado do mapa, os dados são ordenados e o resultado retornado é o ponto mais próximo do usuário.

Figura 15: Função de localização do ponto mais próximo

```

map.js  x
sortByDist
178  function findClosestN(userLocation, numberOfResults, type) {
179    var closest = [];
180    for (var i = 0; i < markers.length; i++) {
181      if(markers[i].type == type) {
182        markers[i].distance = google.maps.geometry.spherical.computeDistanceBetween(userLocation,
183          markers[i].getPosition());
184        markers[i].setMap(null);
185        closest.push(markers[i]);
186      }
187    }
188    closest.sort(sortByDist);
189    return closest.splice(0, numberOfResults);
190  }
191  function sortByDist(a, b) {
192    return (a.distance - b.distance)
193  }

```

Fonte: Autoria Própria

5.7. TESTES

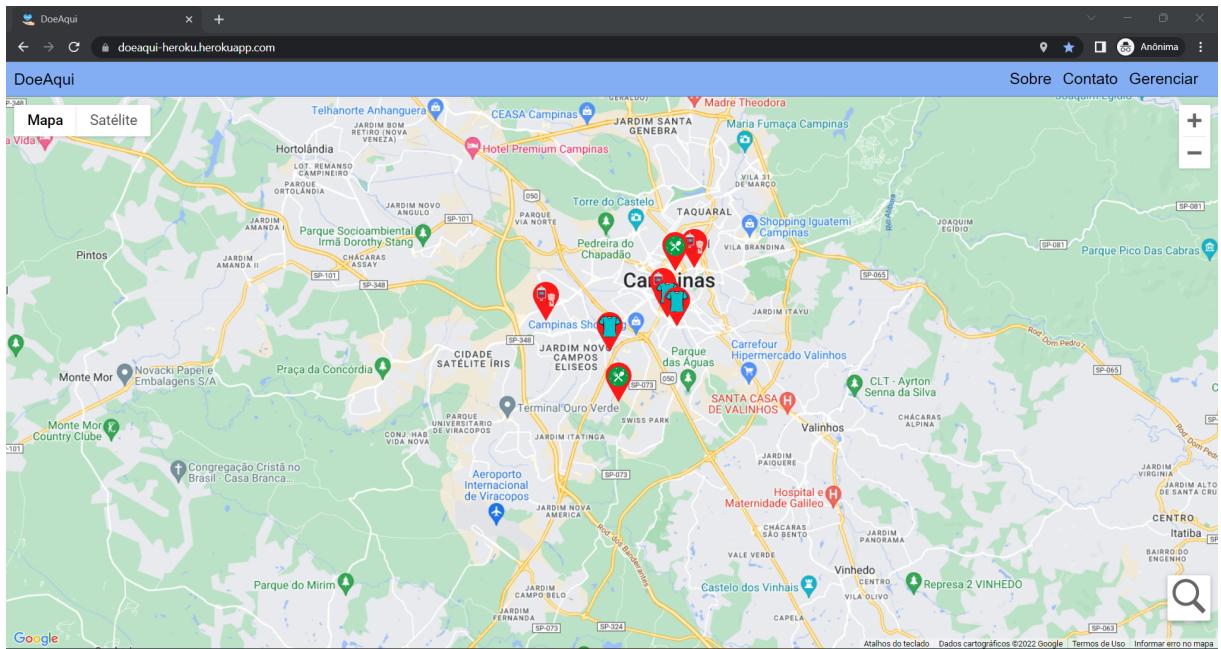
Os testes foram realizados manualmente pelo autor e desenvolvedor do trabalho, sendo assim, não ocorreu uso de nenhuma ferramenta de automatização de testes. Eles foram divididos em duas etapas: testes de caixa-preta e caixa-branca. O teste de caixa-preta simulou usuários sem acesso ao código fonte da aplicação, interagindo com o sistema, seja filtrando os pontos, acessando a visualização dos pontos ou criação de conta. Além disso, outro modelo adotado foi o teste de caixa-branca, abordando um testador tendo acesso ao fluxo dos dados, validando toda a lógica da aplicação e cumprimento das regras de negócio.

Vale destacar que, os testes foram realizados em conjunto com o desenvolvimento, sendo assim, caso ocorresse um erro a funcionalidades era corrigida no mesmo instante e caso não existisse nenhum erro o projeto avançaria para a próxima etapa.

5.8. HOSPEDAGEM DA APLICAÇÃO

A última etapa do projeto é a hospedagem do sistema *web* para o público. Desta maneira, qualquer pessoa que queira realizar ou receber uma doação pode entrar no site e encontrar o centro mais próximo de sua localização, precisando apenas de um aparelho celular ou computador com acesso à internet. A hospedagem foi feita utilizando o Heroku (Figura 16), pois essa plataforma de nuvem fornece uma forma de hospedagem Node.js gratuita, auxiliando com a disponibilidade, escala e infraestrutura da aplicação (HEROKU, 2022).

Figura 16 - Sistema hospedado com Heroku



Fonte: Autoria própria (2022).

6. RESULTADOS

A finalidade deste capítulo é demonstrar os resultados obtidos durante o desenvolvimento do trabalho. O capítulo foi dividido de acordo com cada tela do sistema. A explicação aprofundada está descrita em suas respectivas subseções.

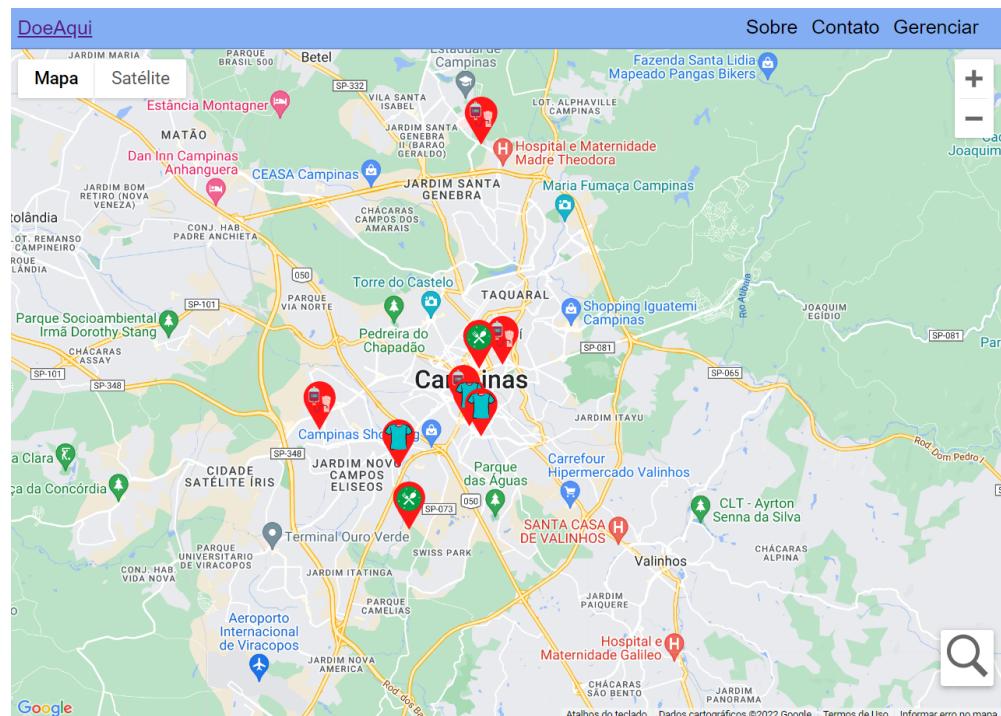
6.1. TELAS DO SISTEMA

Esta seção aborda as principais telas desenvolvidas durante este projeto. Sendo elas: tela inicial do *website*, tela de gerenciamento, tela de visualização do ponto de doação, tela de cadastro do ponto de doação, área de visualização do estoque, tela de controle do estoque, tela de acesso ao gerenciamento e tela de acesso negado.

6.1.1. TELA INICIAL

Ao entrar no site o usuário vai ter a tela inicial do sistema (Figura 17). O cliente tem acesso a todos os pontos de doação cadastrados próximos a sua localização atual. Além disso, é possível filtrar por tipo de doação o ponto mais perto.

Figura 17 - Tela inicial



Fonte: Autoria própria (2022).

6.1.2. TELA DE IDENTIFICAÇÃO

Tela de identificação, responsável por autorizar usuários cadastrados no sistema e negar o acesso de usuários não autorizados à área de gerenciamento (Figura 18).

Figura 18 - Tela de identificação



Fonte: Autoria própria (2022).

6.1.3. TELA DE ACESSO NEGADO

Caso o usuário tente acessar uma área na qual ele não possui privilégios de gerente ele é redirecionado para uma página de negação de acesso (Figura 19).

Figura 19 - Área de acesso negado

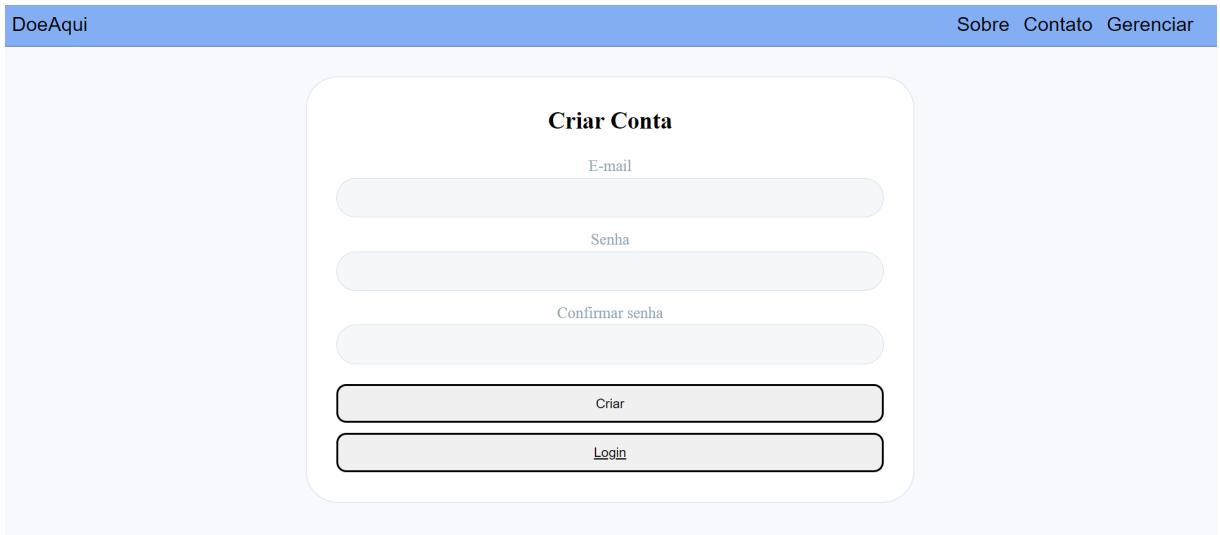


Fonte: Autoria própria (2022).

6.1.4. TELA DE CRIAR CONTA DO GERENTE

Caso o usuário deseje gerenciar centros de doação ele precisará criar uma conta no sistema (Figura 20). Dessa maneira, são solicitados um e-mail, uma senha e a confirmação da senha.

Figura 20 - Tela de criação de conta do gerente



Fonte: Autoria própria (2022).

6.1.5. TELA DE RECUPERAR SENHA

Caso o gerente esqueça a senha da conta, é possível recuperar o acesso na área de esquecimento da senha (Figura 21). Dessa maneira, é solicitado o e-mail da conta e logo em seguida o usuário receberá um e-mail (Figura 22), em sua caixa de mensagens, contendo um link de redirecionamento para uma página de redefinição de senha (Figura 23).

Figura 21 - Tela de recuperar senha



Fonte: Autoria própria (2022).

Figura 22 - E-mail contendo o link de redirecionamento



Fonte: Autoria própria (2022).

Figura 23 - Página de redefinição de senha

Fonte: Autoria própria (2022).

6.1.6. TELA DE GERENCIAMENTO

A área de gerenciamento (Figura 24) permite ao gerente administrar todos os pontos de doação cadastrados por ele mesmo. Deste modo, ele pode criar um novo ponto de doação, editar as informações do ponto, gerenciar o estoque ou até mesmo excluir o ponto de doação.

Figura 24 - Tela de gerenciamento

Fonte: Autoria própria (2022).

6.1.7. TELA DE VISUALIZAÇÃO DO PONTO DE DOAÇÃO

A área de visualização das informações do ponto de doação (Figura 25), exibe os dados cadastrados, como por exemplo: nome, história, site, meios de contatos, localização no mapa e estoque do local.

Figura 25 - Tela de visualização do ponto de doação

Fonte: Autoria própria (2022).

6.1.8. TELA DE CRIAÇÃO DO PONTO DE DOAÇÃO

A área de criação do ponto de doação (Figura 26) só está disponível aos gerentes cadastrados no sistema.

Figura 26 - Tela de criação do ponto de doação

The screenshot shows a web-based form for creating a donation point. At the top left is a blue button labeled 'DoeAqui'. At the top right are links for 'Sobre', 'Contato', and 'Gerenciar'. The main form area has a title 'Dados' at the top. It contains several input fields:

- 'Nome' (Name) with placeholder text 'Nome'.
- 'Sobre' (About) with a large text area placeholder 'Sobre'.
- 'Email de contato' (Contact email) with placeholder text 'centro@gmail.com'.
- 'Link do site' (Site link) with placeholder text 'https://'.
- 'Número do Whatsapp' (WhatsApp number) with placeholder text '(00) 9 0000-0000'.
- 'CEP' (ZIP code) with placeholder text '00000-000'.

A green 'Buscar' (Search) button is located below these fields. Below the button is a map from Google Maps showing a location in São Paulo, Brazil. The map includes labels for 'CONTINENTAL', 'EE Professora Maria Julieta de Godói...', 'Furacão Distribuidora de Peças Automotivas', 'Makro Atacadista', 'Rodovia Presidente Dutra (SP-073)', 'Cipola - Gráfica e...', 'JARDIM DO LAGO II', and 'FITPECAS - Distribuidor de Veículos Ltda.'. A red marker is placed on the map near the Furacão Distribuidora address. A small 'Selecionar no mapa' (Select on map) button is also visible on the map interface.

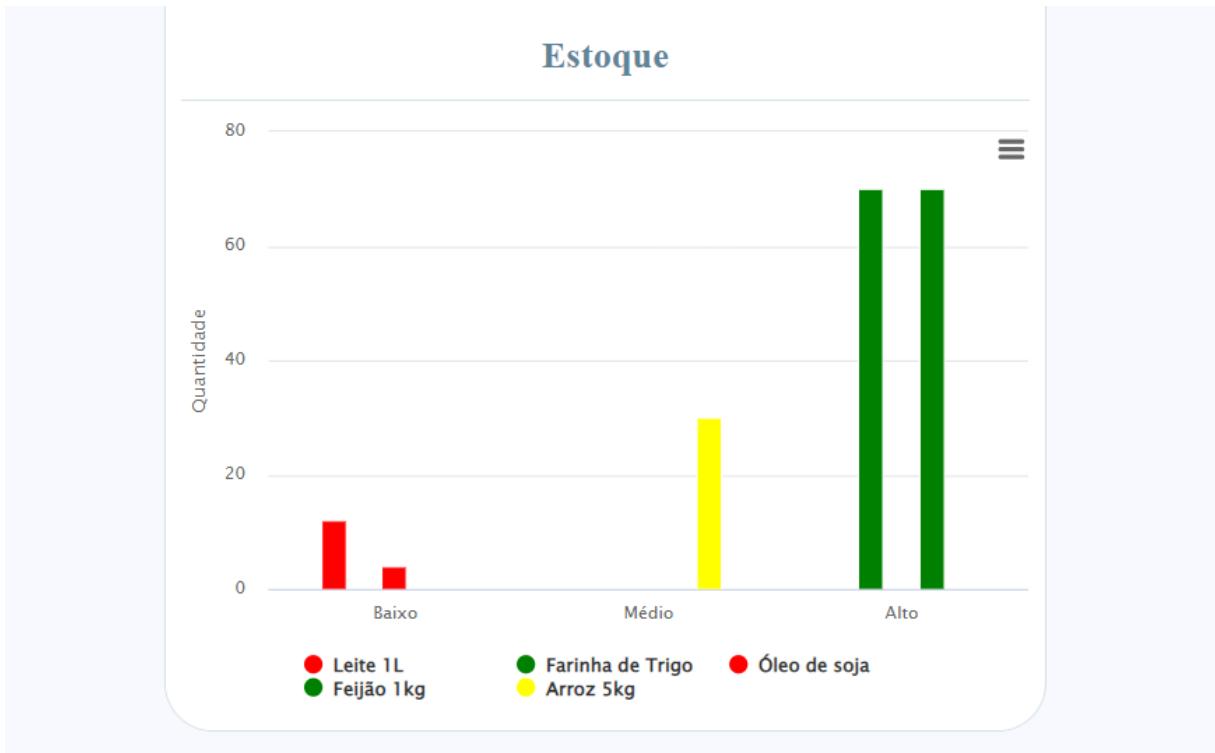
Voltar

Fonte: Autoria própria (2022).

6.1.9. TELA DE VISUALIZAÇÃO DO ESTOQUE

A visualização do estoque (Figura 27), está presente na tela de visualização do ponto de doação (Figura 25).

Figura 27 - Tela de visualização do estoque



Fonte: Autoria própria (2022).

6.1.10. TELA DE CONTROLE DO ESTOQUE

A tela de controle do estoque (Figura 28), só está disponível após a criação do ponto de doação, nessa tela é possível definir quais são os produtos doados, quantidade disponível e nível do estoque.

Figura 28 - Tela de controle do estoque

Item	Quantidade	Status
Arroz 5kg	30	Médio
Feijão 1kg	70	Alto
Óleo de soja	4	Baixo
Leite 1L	12	Baixo
Farinha de Trigo	70	Alto

Fonte: Autoria própria (2022).

6.1.11. TELA DE CONFIGURAÇÕES DO PERFIL DO GERENTE

A tela de configuração do perfil (Figura 29) é disponibilizada apenas para usuários cadastrados no sistema, responsável por editar as informações exibidas na tela de gerenciamento (Figura 24), como por exemplo, nome e foto.

Figura 29 - Tela de configurações do perfil

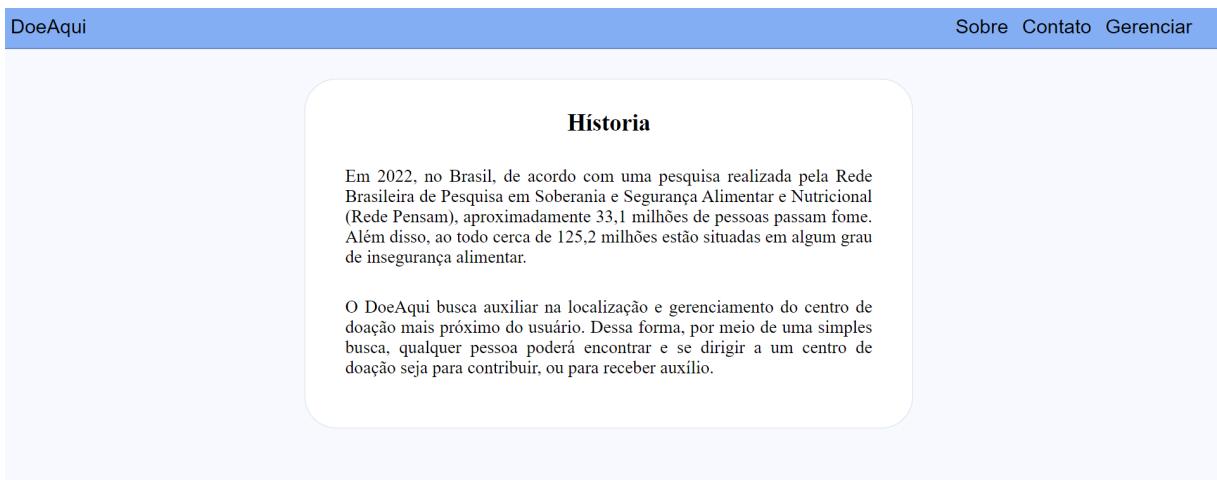


Fonte: Autoria própria (2022).

6.1.12. TELA DE HISTÓRIA DO WEBSITE

Essa área é responsável por contar a motivação por trás da criação do website DoeAqui (Figura 30).

Figura 30 - Tela de história do website

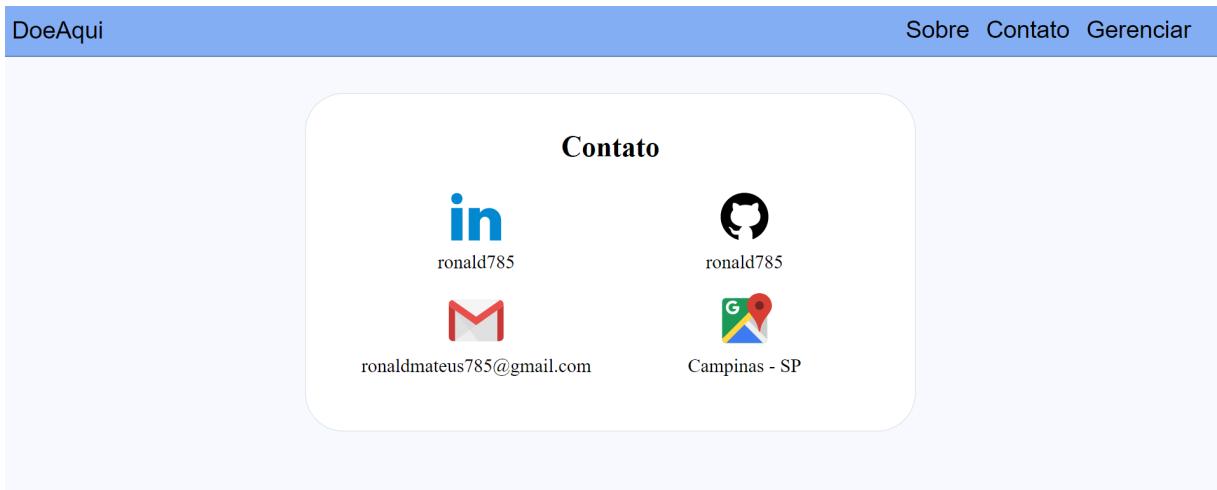


Fonte: Autoria própria (2022).

6.1.13. TELA DE INFORMAÇÕES DE CONTATO

Essa área é responsável por fornecer as maneiras de entrar em contato com o desenvolvedor do sistema (Figura 31).

Figura 31 - Tela de informações de contato



Fonte: Autoria própria (2022).

7. CONSIDERAÇÕES FINAIS

O agravamento do nível da fome na pandemia provocado pela covid-19, doença causada pelo coronavírus SARS-CoV-2, foi o motor para o desejo de propor um sistema de unificação dos centros de doações. Dessa forma, primeiramente foi necessário analisar o cenário atual da população.

Tendo em vista esse cenário, foi desenvolvida uma aplicação *web* que permite localizar o centro mais próximo do usuário e gerenciar os centros de distribuições cadastrados.

Observando os objetivos específicos listados na seção 3.2 do documento, todos eles foram cumpridos, tanto o cadastramento do centro, rotinas para o gerente controlar os centros e implementação do Google Maps API.

Além do mais, a hospedagem utilizando a plataforma Heroku, tornou-se uma ótima solução para disponibilizar o sistema para qualquer usuário com acesso à internet.

Por último, o sistema pode receber novas atualizações, como por exemplo, a integração de um sistema de chatbot para auxiliar novos usuários, proporcionando uma forma de tirar dúvidas e facilitar a interação com o DoeAqui. Outrossim, o site também pode ser convertido em um aplicativo móvel, provendo uma melhor experiência aos usuários de dispositivos móveis.

REFERÊNCIAS

ANDRADE, A. **O que é Template Engine?** , 2020. Disponível em: <<https://www.treinaweb.com.br/blog/o-que-e-template-engine>>. Acesso em: 20 out. 2022

ANDRADE, A. **O que é Firebase? | Blog TreinaWeb**. Disponível em: <<https://www.treinaweb.com.br/blog/o-que-e-firebase>>. Acesso em: 20 jun. 2022.

BARTIE, A. **Garantia da qualidade de software: Adquirindo Maturidade Organizacional**. 1^a edição ed. [s.l.] GEN LTC, 2002.

BERNERS-LEE, T.; MASINTER, L. M.; MCCAHILL, M. P. **Uniform Resource Locators (URL)**. [s.l.] Internet Engineering Task Force, dez. 1994. Disponível em: <<https://datatracker.ietf.org/doc/rfc1738>>. Acesso em: 27 nov. 2022.

CASA DA SOPA. **Casa da Sopa – Campinas**. , 2022. Disponível em: <<https://casadasopacampinas.com.br/>>. Acesso em: 27 jun. 2022

CEASA. **CEASA Campinas**. Disponível em: <<http://www.ceasacampinas.com.br/>>. Acesso em: 27 jun. 2022.

CLARK, J. **Firebase vs. Firestore | Quais são as diferenças?** Disponível em: <<https://blog.back4app.com/pt/firebase-vs-firestore-quais-sao-as-diferencias>>. Acesso em: 20 jun. 2022.

COVO, K. **Configurando o Nunjucks**. Medium, 4 maio 2019. Disponível em: <<https://medium.com/@kaiquecovo/configurando-o-nunjucks-5333fee34c5b>>. Acesso em: 20 out. 2022

DIONISIO, E. **Introdução ao Visual Studio Code**. Disponível em: <<https://www.devmedia.com.br/introducao-ao-visual-studio-code/34418>>. Acesso em: 20 jun. 2022.

FEAC. **Fundação FEAC » Empoderamos pessoas - Impulsionamos organizações - Potencializamos territórios**. Disponível em: <<https://feac.org.br/>>. Acesso em: 27 jun. 2022.

FIREBASE. **Firebase Authentication**. Disponível em: <<https://firebase.google.com/docs/auth?hl=pt-br>>. Acesso em: 20 out. 2022.

FLATSCHART, F. **HTML 5: embarque imediato**. 1^a edição ed. Rio de Janeiro: Brasport, 2011.

GOOGLE. **Plataforma Google Maps**. Disponível em: <<https://developers.google.com/maps?hl=pt-br>>. Acesso em: 20 out. 2022.

GUEDES, M. **Git e GitHub: quais as diferenças?** Disponível em: <<https://www.treinaweb.com.br/blog/git-e-github-quais-as-diferencias>>. Acesso em: 20 jun. 2022.

GUITARRARA, P. **Meridiano de Greenwich: função, localização**. Disponível em: <<https://mundoeducacao.uol.com.br/geografia/meridiano-greenwich.htm>>. Acesso em: 19 out. 2022.

HARADA, E. **O que é Figma e como você pode usufruir dessa ferramenta de design - TecMundo**. Disponível em: <<https://www.tecmundo.com.br/software/236320-figma-voce-usufruir-dessa-ferramenta-de-design.htm>>. Acesso em: 20 jun. 2022.

HEROKU. **Platform as a Service | Heroku**. Disponível em: <<https://www.heroku.com/platform>>. Acesso em: 20 out. 2022.

INTHURN, C. **Qualidade E Testes De Software**. [s.l.] VISUAL BOOKS, 2001.

KUROSE, J.; ROSS, K. **Redes de Computadores e a Internet: Uma Abordagem Top-Down**. 6ª edição ed. São Paulo: Pearson Universidades, 2013.

NIELSEN, J. **Projetando Websites**. 6ª edição ed. [s.l.] Campus, 2000.

OLIVEIRA, M. **Coordenadas Geográficas do Brasil - pontos extremos e áreas**. Disponível em: <https://www.suapesquisa.com/geografia/coordenadas_geograficas.htm>. Acesso em: 19 out. 2022.

REDE PENSSAN. **Insegurança alimentar e Covid-19 no Brasil: VIGISAN, Inquérito Nacional sobre Insegurança Alimentar no Contexto da Pandemia da Covid-19 no Brasil**. [s.l: s.n.]. Disponível em: <<https://olheparaafome.com.br/wp-content/uploads/2022/06/Relatorio-II-VIGISAN-2022.pdf>>. Acesso em: 20 set. 2022.

SOMMERVILLE, I. **Engenharia de Software**. 10ª edição ed. [s.l.] Pearson Universidades, 2019.

SOUTO, M. **O que é front-end e back-end?** Disponível em: <<https://www.alura.com.br/artigos/o-que-e-front-end-e-back-end>>. Acesso em: 27 jun. 2022.

ZANETTE, A. **Framework x Biblioteca x API. Entenda as diferenças!** BeCode, 30 maio 2017. Disponível em: <<https://becode.com.br/framework-biblioteca-api-entenda-as-diferencias/>>. Acesso em: 19 out. 2022